WANG

VS Languages

LANGUAGES ON THE WANG VS

The versatile Wang VS system supports multiple high-level programming languages and a sophisticated set of software development tools for the applications programmer. The VS languages are based on industry standards and include Wang extensions that provide interactive programming capabilities. Because the VS has the ability to call programs written in other languages, the system designer can write in the language that is best suited to a specific task.

The primary languages supported by the Wang VS for commercial applications development are COBOL and RPG II. In addition to business-oriented languages, the VS also supports PL/I, FORTRAN, and Assembler Language. VS Procedure language and a high-level BASIC complete the VS language set.

Interactive program development tools enhance all VS languages. These tools facilitate development, debugging, and testing to provide the complete applications programming environment.

Language Features

An easy-to-use operating system supports all VS programming languages. Currently available VS languages have been extended to include features that solicit and receive user-entered data from the interactive workstation.

All VS languages provide automatic interface with the *VS Data Management System (DMS).* The DMS facilities provide many features common to all five languages.

Data files are accessed by a specific key value or in consecutive, record-by-record storage sequence. Data files produced by programs written in one language can be accessed by programs written in any VS language.

Indexed files are accessed sequentially along a primary key path or randomly by specific key value. Sixteen alternate key paths are provided for COBOL, RPG II, BASIC and Assembler language. *Consecutive files* are accessed sequentially or randomly by relative record number.

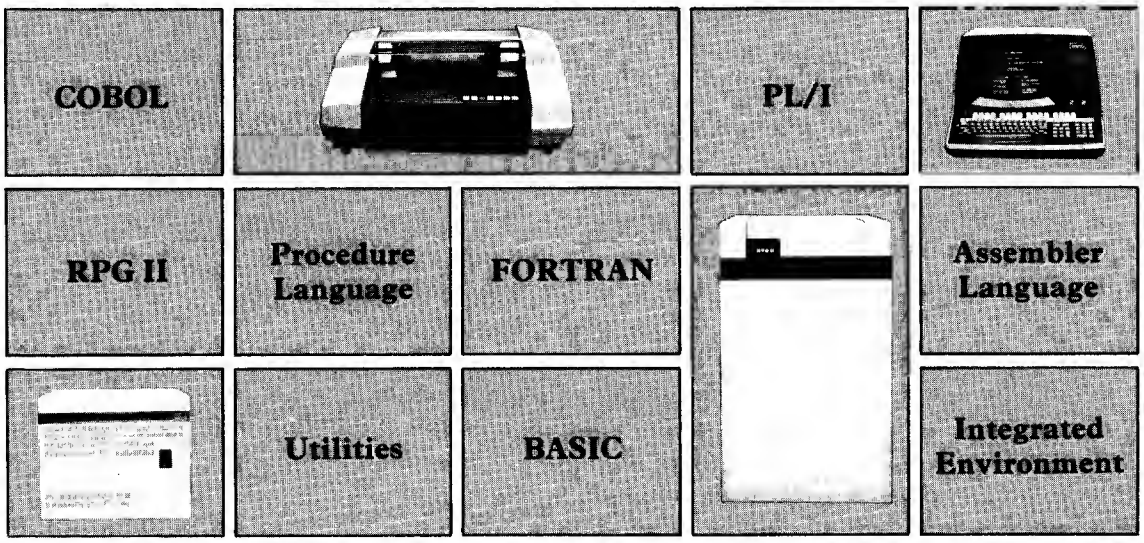
The system provides *three record formats:* fixed-length, variable-length, and compressed.

Record compression conserves disk storage by representing strings of 3 to 128 characters in 2 bytes. Handling of compressed records is automatic.

Shared file processing of indexed data files permits multiple updating. While one user updates a file, another user can simultaneously run a program against it.

Anticipatory buffer priming, a technique for reading data into a buffer before it is needed for processing, reduces the I/O wait for data.

The *large buffering technique* and *buffer pooling* are additional buffering techniques available on the VS.



VS Environment Features

The VS Operating System has many features that aid the programmer in making efficient use of its varied resources. Some of these features are described below.

Programs written in any of the supported languages can call *subprograms* written in any other language. The programmer can implement functional portions of a system design in the language that is best suited to the application.

More than one user can automatically share program files. *Program sharing* enables many users to execute the same program at the same time without requiring multiple copies in memory.

For increased flexibility, VS languages incorporate disk, tape, printer, and workstation *file support.*

Communications and emulation software allow the VS to communicate with host computers, compatible terminals, and other Wang systems, thus supporting a variety of industry-standard protocols.

The system allows *background processing.* Background jobs submitted from the workstation require no additional operator intervention, thus freeing the workstation for interactive jobs.

Print spooling optimizes printer usage while reducing processing time. It sends printer output to a disk file rather than directly to a printer. Print files are produced at the faster disk I/O speed and are not limited by printer availability or I/O speed.

Security on the VS protects program and data files, the system, and devices. Each file and user is assigned access rights; access is limited to users with these rights.

The *linking* capability provides interprogram communication between programs of the same or different languages.

The internal character set for all languages and data files is *ASCII.*

A *programmer-oriented environment* supports all programming languages.

COBOL

Wang VS COBOL, in conformance with the Federal Information Processing Standard (FIPS), includes all features specified for *ANSI (X3.23-1974) Level 1 COBOL.* Features in these modules include: Nucleus, Table Handling, Sequential I/O, Indexed I/O, Library, Debug, and Interprogram communication. Wang VS COBOL also includes many useful Level 2 features.

COBOL is used extensively in business applications requiring large file handling and output formatting. It is suitable for maintaining current information needed by inventory, billing, payroll, and other general-purpose business applications.

With its interactive screen formatting extensions, VS COBOL also provides an on-line inquiry and update capability that is not available in most standard versions of the language. This additional capacity to display, read, and modify record information during program execution enables the programmer to design a responsive, accurate system with reduced processing time. Each display can serve as a record containing many fields that can be interactively interrogated and updated. In addition to the general features available for all VS programming languages, VS COBOL offers a variety of unique processing capabilities.

A single statement controls *workstation file I/O* for displaying data on the screen, for reading and verifying data input at the workstation keyboard, and for transferring the data to a storage file.

The programmer can selectively transfer *data fields* between the screen and the working files.

Each field can have condition and result indicators set, be tested for display status information, and be displayed with different attribute characteristics.

The *extended data transfer* capability converts and moves free-form character string representations of numbers to fields that can be used for computation.

Advanced file sharing capabilities allow multiple users to update or retrieve shared resources. A user can hold any combination of shared records and files without interference from other users.

Special options that permit *program tuning* for more efficient use of memory and external storage are available.

RPG II

Originally developed as a special-purpose language for generating reports, Wang VS RPG II has been continuously enhanced and extended,

and is now a general-purpose business programming language. In addition to a versatile report creation capability, Wang VS RPG II can perform looping, testing, branching, computations, and data and file manipulation.

To make interactive program development easier, VS RPG II now includes a workstation specification form, which enables the programmer to control *data entry* and *data display* at the workstation.

VS RPG II programs can use the *Program Function (PF) keys* to accept user input. *Special indicators* and *global variables* monitor PF key responses and condition calculation or output specifications. Data can be validated upon entry and errors brought to the user's attention by blinking fields or the workstation alarm.

VS RPG II supports the VS *advanced sharing* facilities. A user can hold any combination of shared records and files for update, allowing an entire business transaction to be posted without interference from other users.

VS RPG II provides powerful *automated conversion aids* which convert applications from System/3, System/32, and System/34 RPG II. VS RPG II programs can call *external subroutines* written in any supported language.

PL/I

Wang VS PL/I is an optimization of the *G subset* language adopted by the American National Standards Institute (ANSI) in 1981. While VS PL/I and the G subset are easier to learn and more efficient than the full language, they still offer the most useful PL/I features. These include block structure, a rich set of data types, flexible I/O constructs, and programmer control of exception handling. A description of other PL/I features follows.

Three classes of storage (automatic, static, and based) allow the programmer to store data in the manner that is best suited to each variable in a particular application.

Easy-to-use, list-directed and edit-directed *stream I/O* facilitates I/O specification.

Record I/O processes entire records and offers complete programmer control over output data format. Record I/O is supported for tape, printer, and disk operations.

Four *arithmetic data types* are available for numeric calculations: fixed and floating point binary, and fixed and floating point decimal. A special *picture data type* represents certain arithmetic values as character string values. *Character and bit string data types* facilitate string manipulation.

When compiling, the programmer can alter program text through the use of *special statements.* These include format specifications for compiled source listings, text insertion, and text replacement. Text files from any library can be included in the program text.

FORTRAN

FORTRAN is a programming language that is well suited to scientific and problem-solving applications that require a high volume of efficient numeric processing. This language offers built-in mathematical functions, record-oriented I/O, and conversion of internal values to and from their external representations. FORTRAN also has the functional capability to process integer, real (floating-point), single-precision, double-precision, complex, and logical data types. FORTRAN also provides statements for assignment, program control, I/O, formatting, data initialization, data specification, statement function definition, and subprogram definition.

VS FORTRAN is ANSI FORTRAN 66 (X3.9-1966) extended with Wang features. It allows arithmetic statement functions and function and subroutine subprograms. FORTRAN programs can call subroutines written in VS COBOL, BASIC, RPG II, and Assembler.

VS FORTRAN includes such intrinsic and extrinsic functions as logarithmic and trigonometric operations, absolute value, and square root functions. Additional features are as follows.

The *COPY statement* allows the user to include source lines from another file.

The user can specify an *end-of-file exit* on the READ statement and *error exits* for READ and WRITE.

Expressions are allowed in DO statement parameters and subscript quantities.

Arrays can contain up to seven dimensions.

Hexadecimal data can be read and written.

The user can enclose *character strings* in quotes. This method is generally more convenient than the Hollerith constant specification (nH).

The *IMPLICIT statement* allows the user to override the default data type associated with the initial character of a variable name.

Default word sizes for integer, real, and logical variables can be explicitly changed.

ASSEMBLER LANGUAGE

VS Assembler language, a convenient tool for programming at the machine instruction level, is compatible with the 360/370 Assembler language. The Assembler language offers a comprehensive, general-purpose instruction set that provides access to all elementary functions of the VS computer.

Assembler instructions are variable-length, ranging from one halfword (two bytes) to four halfwords (eight bytes) in length. Besides the standard *RR, RX, RS, SI, S, SS,* and *SSI formats,* the Wang VS Assembler provides two other formats:

RL Format — Register and relative address operation (two halfwords)
RRL Format — Register-pair and relative address operation (two halfwords)

With VS Assembler, the user can define *macros,* a series of assembler language statements. A single macro instruction statement is placed in a program wherever the associated assembler statements are needed. When the assembler encounters a macro instruction, it automatically generates the corresponding macro routine, that is, the sequence of statements in the macro definition. Macros are particularly useful for routines that are used repeatedly or for standardizing common routines used in different applications.

Wang provides a number of system macros to perform common tasks for the assembler programmer, including soliciting and validating user input from the workstation, and directly accessing system information. In addition, the VS Assembler language includes the features described below.

Special instructions permit *stack and queue manipulation;* address arithmetic permits manipulation of main storage addresses.

Floating-point binary and *floating-point decimal* arithmetic instructions are provided, together with *logical and string processing* instructions.

The language provides *data conversion, manipulation, and editing* instructions. It also includes an extensive set of *branching and testing* instructions with extended mnemonics.

BASIC

Wang VS BASIC is an extensively enhanced version of the original Dartmouth BASIC. VS BASIC preserves the clarity and ease-of-use of the BASIC language, while adding significantly to its power and flexibility.

Because of its straightforward syntax and simple instruction set, BASIC is a preferred language in an instructional environment. A number of special features enhance the interactive design of VS BASIC. These include screen-formatting and automatic data validation and correction, which facilitate conversational use of the workstation by application users, as well as the functions described below.

The programmer has access to a full set of *numeric functions,* including arithmetic, matrix arithmetic, trigonometric, exponential, transcendental, and programmer-defined functions. Both integer and real (floating point) computations are available.

VS BASIC provides a complete set of *matrix arithmetic* instructions, including guidelines for performing matrix inversion and matrix multiplication.

With a single statement, the user can perform *internal matrix sorting,* in ascending or descending order.

One- and two-dimensional alphanumeric and numeric *arrays* are allowed.

Comprehensive alphanumeric *string and bit manipulation* facilities are provided.

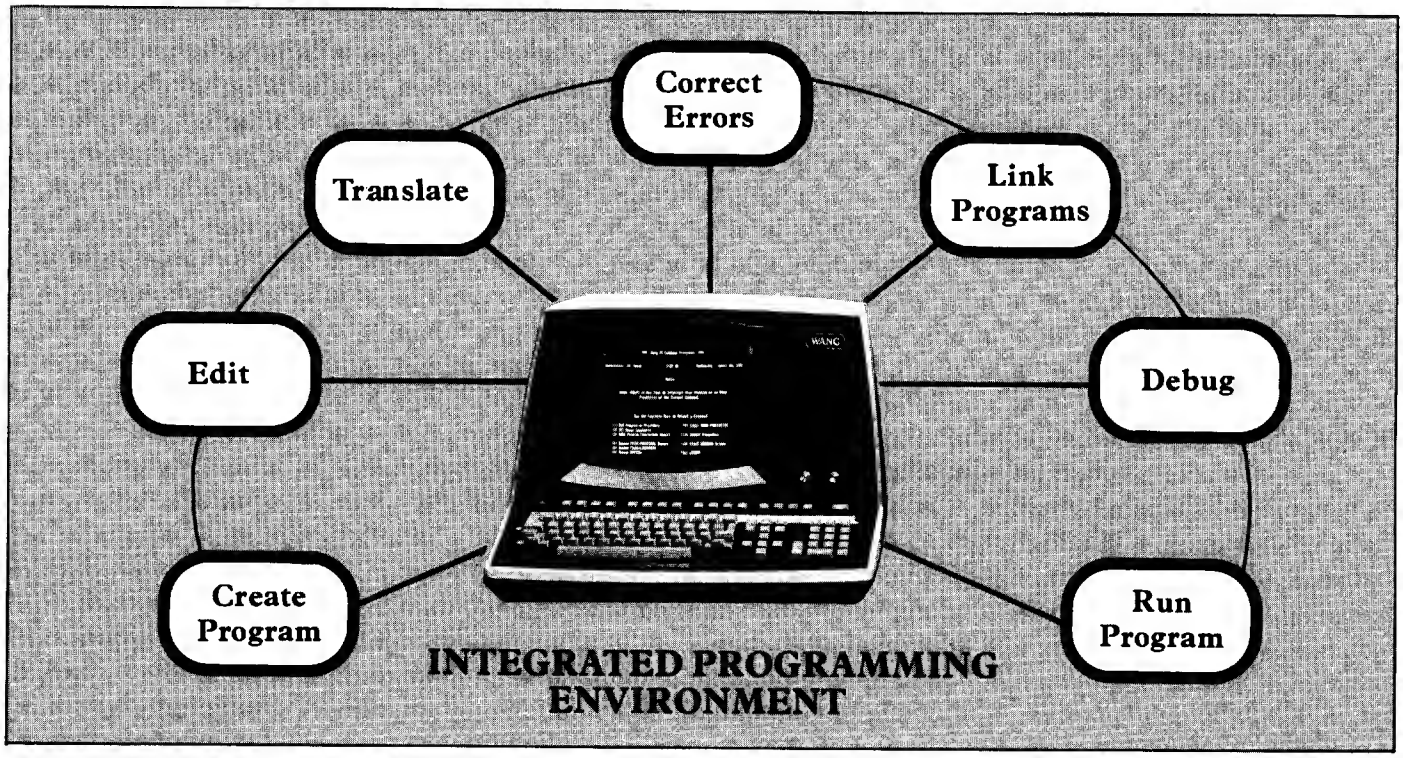
Variable names can contain up to 64 characters.

Labels of up to 64 characters identify BASIC statements. All program control transfers can be directed to alphanumeric labels, as well as to line numbers.

The program *comment facility* includes two types of documentation remarks: those written within statements and those written as separate statements.

An added compiler option produces a sorted *cross-reference listing.*

A complete set of *Boolean logic functions* is provided.



PROCEDURE LANGUAGE

The VS Procedure language combines the capabilities of a *command language* and a *job control language* without the complex syntax normally associated with these types. Procedures are special text files executed by the Procedure Interpreter. They can run many of the operations normally performed interactively by a user at a workstation (e.g., running programs, supplying runtime parameters to a program, mounting and dismounting volumes, scratching, protecting, or renaming files), extract information from the system, and submit and print jobs.

In addition, Procedure language includes *testing and branching instructions.* These permit conditional execution of one or more procedure statements based on the return code generated by a previously executed statement. Additional VS Procedure language features are described below.

Programs can be run and runtime parameter information can be specified without user involvement. Procedures *reduce the number of keystrokes* required to run an application program, thus avoiding possible runtime errors.

Without intervening, a user can *run several programs automatically* and in sequence.

Free-form syntax and English-like statements make procedures *easy to write and understand,* while providing all command control needed for program execution.

The Procedure Interpreter permits *forward and backward branching, nesting of procedures,* and *backward referencing* to obtain parameter values specified in a previously executed procedure statement.

PROGRAM DEVELOPMENT TOOLS

For program development purposes, the VS interactive system offers a number of features not found in traditional, batch-oriented systems. Included among these features are a variety of program development tools that are useful in any language.

Of special interest to the programmer are several VS utilities that provide an *integrated approach* to program development from editing to translating, linking, and debugging.

WANG

ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851
TEL.(617)459-5000
TWX 710-343-6769, TELEX 94-7421

Integrated Text Editor

The VS Editor integrates all the functions needed to *create, edit, compile, link, and run programs* on the VS. Using this utility, the programmer enters and edits program text, invokes the appropriate translator (compiler or assembler), displays translator-generated error messages, re-edits and retranslates as needed, and, finally, executes the program. This edit-translate-execute cycle can be repeated as often as necessary within the Editor until all errors have been identified and corrected.

Additional Editor capabilities include character and line insertion and deletion, global replace and scanning functions, and the ability to move specified text strings around in the file. An external copy function permits text from other source program files to be copied into the current program at a specified point. The Editor also offers functions that inspect programs page by page, search for specified text strings, or position the cursor to a specified file line.

EZFORMAT Utility

The EZFORMAT utility *designs screen displays* and *generates the source code* necessary to reproduce them. The user places the code created in the screen definition area of a COBOL, RPG II, or BASIC language source code, or in the static section of an Assembler program. With EZFORMAT, the user can also create a data entry routine with a defined screen format, or create a *customized menu display* that supplements or replaces the standard system menu for a particular application.

Translator Error Messages

All VS translators (compilers, procedure interpreter, and the assembler) perform *exhaustive syntax analysis* and produce a list of error messages when a translation is completed. Notification of errors at translation time allows the user to correct the source program (using the Editor) and then retranslate the corrected program before attempting to execute it.

Interactive Debug Processor

To identify and correct such runtime program errors as improper branching and infinite loops, the VS provides an Interactive Debug Processor. Because runtime errors are often difficult to identify, the Debug Processor permits the programmer to *inspect program code* and to *inspect and modify data.*

The VS Debug Processor has an additional *symbolic debugging* feature that displays the current section of the source program at the workstation. It also permits high-level language data values to be examined and modified by symbolic data name rather than address. The Symbolic Debug screen displays the runtime error message with the statement number, verb, Program Control Word (PCW), and section of source code currently being executed at interrupt time. From this screen, the programmer might select any of the debug options. The Debug Processor includes facilities for examining and modifying the General Registers, the Program Control Word, and data in memory. It also provides for setting breakpoints and stepping through program execution.

After debugging is complete, the programmer can invoke an optional *optimization phase* to remove debug data, thereby producing more efficient object code.

VS Linker

By supplying the names of the programs to be linked, the programmer can invoke the VS Linker utility which joins into a single program any number of *programs written in different languages.* Because each language has features that make it particularly useful for certain jobs, the programmer might elect to use several languages within a particular application. VS language compilers can communicate with programs written in other languages and can pass parameter information to the external programs.

VS INFO FACILITY

VS INFO can be used to view Wang reference manuals on-line or to display relevant help text while using a Wang software product. VS INFO allows the user to mark the current section with an "electronic placemarker" for easy return after following up references to other sections. VS INFO also allows a split-screen display for uses such as parallel display of a table of contents and the corresponding text. Users can skim the text sequentially, scroll forward and backward, and search for key words or section numbers.

Wang Laboratories, Inc., reserves the right to change specifications without prior notice.

Printed in U.S.A.
800-2201-07
9-82-50M